Haaga-Helia
ammattikorkeakoulu Oy

Ohjelmistokehityksen menetelmiä alakouluun – Digitaalinen oppimateriaali vuosiluokille 3–6

Petri Santala

Opinnäytetyö
Tietojenkäsittelyn koulutusohjelma
2018

Tiivistelmä

Haaga-Helia ammattikorkeakoulu Oy

Tekijä
Petri Santala

Koulutusohjelma
Tietojenkäsittelyn koulutusohjelma

Raportin/Opinnäytetyön nimi	**Sivu- ja liitesivumäärä**
Ohjelmistokehityksen menetelmiä alakouluun – Digitaalinen oppimateriaali vuosiluokille 3–6	21 + 5

Tämän opinnäytetyön tavoitteena oli luoda oppimateriaali, joka tutustuttaa oppilaita ohjelmistokehitykseen. Tavoitteen perusteluna käytän sitä, että nykyaikaiset ohjelmat eivät synny vain ohjelmoijan työn tuloksena, vaan ohjelmia suunnitellaan ja kehitetään monialaisissa tiimeissä. Ohjelmoinnin opetukseen on olemassa paljon materiaalia, mutta mielestäni ohjelmoinnin opetus pitäisi tuoda enemmän ohjelmistokehityksen kontekstiin.

Opinnäytetyön toiminnallisen osan ohjelmointiympäristöksi valitsin Scratch-ohjelmointiympäristön, koska se visuaalisen käyttöliittymänsä vuoksi soveltuu alakoulun käyttöön. Scratch on laajassa käytössä ja tuttu. Scratchin peruskäyttö on helppoa, mutta sillä voi silti tehdä monipuolisia ja laajoja ohjelmia. Opinnäytetyöni yhtenä tavoitteena on osoittaa, että Scratch taipuu moneen.

Opinnäytetyön toiminnallisen osan suunnittelun lähtökohtana oli Legon tapa rakentaa osista koostuvia monipuolisia kokonaisuuksia. Samoin kuin Legoissa, myös Scratchissa ohjelmakokonaisuus rakennetaan yhdistelemällä paloja. Lego-ideologia toistuu myös tehtävävihkon tasolla. Tehtävävihkossa annetaan oppilaille vaiheittainen ohjeistus, jonka avulla oppilas voi itsenäisesti koostaa ohjelmakokonaisuuden. Taustalla on ajatus siitä, että ohjelmointia oppii ensin mallista kopioimalla ja valmista koodia muuntelemalla.

Oppimateriaalin tarkoitus on tutustuttaa oppilas ohjelmistokehityksen prosessin eri rooleihin, tehtäviin ja työkaluihin. Työkaluiksi on valittu helppokäyttöisiä ja ilmaisia ohjelmia, joiden avulla oppilas voi muunnella ohjelmaa.

Opinnäytetyön tuloksena syntyi tuote, josta aion kerätä palautetta jatkokehitystä varten.

Asiasanat
Ohjelmistokehitys, suunnittelu, ohjelmointi, opetussuunnitelma, Scratch, visuaalinen ohjelmointikieli, koodaus

Sisällys

1 Johdanto

Tämän opinnäytetyön aiheen innoittajana toimi eräs Facebookin Tieto- ja viestintätekniikka opetuksessa/ICT in Education -ryhmään kirjoitettu viesti. Viestissä kysyttiin tietävätkö ryhmän jäsenet työpajatyyppistä laajaa ohjelmointitehtävää, jossa oppilas voisi edetä omaa tahtiaan. Scratch oli viestin kirjoittajan mukaan jo "nähty" ja materiaalin toivottiin olevan ilmainen. Jäin pohtimaan mitkä asiat vaikuttavat siihen, että Scratch-ohjelmointiympäristö ei riitä opettajan ja oppilaiden tarpeisiin, joten päätin laatia laajan, työpajatyyppisen oppimateriaalin Scratchilla.

Opinnäytetyössä pohditaan ohjelmoinnin opettamisen haasteita alakoulussa ja luonnostellaan ohjelmoinnin opetukseen soveltuvaa oppimateriaalia. Oppimateriaalin luonnostelun lähtökohtana on ollut mallista oppiminen, Lego-ideologia ja oppilaan itseohjautuvuus. Tämän opinnäytetyön tarkoituksena ei ole testata näitä pedagogisia ratkaisuja tieteellisin menetelmin. Kurssimateriaali on suunniteltu ohjelmistokehityksen näkökulmasta ja se on pyritty tekemään siten, että se ohjaa oppilasta etenemään ohjelman ohjelmoinnin alusta loppuun siten, että oppilaalle on koko ajan selvää mitä pitää tehdä seuraavaksi. Kokonaisuus koostuu pienistä ohjelmoitavista osista, jotka liitetään yhteen, jolloin saadaan yksi toimiva kokonaisuus. Yhtenäisessä tehtävänannossa on pyritty käsittelemään ohjelmistokehityksen ja ohjelmoinnin eri osa-alueita ja toiminnallisuuksia laajasti. Ratkaistavat tehtävät ovat helppoja ja ohjelmoitavan kokonaisuuden aihe oppilaalle ennestään tuttu. Oppilaan ei tarvitse käyttää energiaa ymmärtääkseen, mitä asioita ohjelmalla ratkaistaan. Oppilas voi keskittyä siihen, miten tehtäväkokonaisuuden eri toiminnallisuudet toteutetaan. Tavoitteena on synnyttää onnistumisen kokemuksia ja oivaltamisen riemua sekä herättää kiinnostusta ongelmien ratkaisemiseen.

Opinnäytetyön toiminnallisen osan muodostaa tuotekehitysprosessin lopputuotteena syntynyt oppimateriaali. Oppimateriaalin tarkoituksena on toimia ponnahduslautana sekä kiinnostuksen herättäjänä ohjelmoinnin opiskeluun. Opinnäytetyö sisältää kirjallisen ja toiminnallisen osan. Toiminnallinen osa on kehitetty Scratch-ohjelmointiympäristöä apuna käyttäen ja materiaalin tarkoitus on esitellä oppilaille kattavasti ohjelmoinnin ominaisuuksia ja rakenteita. Oppimateriaalissa ei opeteta Scratch-ohjelmointiympäristön käyttöä eikä Scratch-ohjelmointia. Materiaali on kytketty työelämän haasteisiin ja oppilaiden arkeen. Kokonaisuuden on tarkoitus tukea oppilaan itseohjautuvuutta sekä toimia onnistumisten mahdollistajana. Oppilas voi suorittaa kokonaisuuden missä tahansa, yksin tai ryhmässä, eikä hän välttämättä tarvitse opettajan ohjausta ohjelman valmiiksi saattamiseksi. Kun oppilas on saanut ohjelman valmiiksi, hän voi auttaa muita ongelmatilanteissa. Hän voi myös tehdä luovia versioita ohjelmasta.

Yksi opinnäytetyön suunnittelun lähtökohdista oli laatia materiaali, joka havainnollistaa oppilaille ohjelmistokehitysprosessia eikä pelkästään ohjelmointia. Ajatuksena taustalla on, että innostuminen ohjelmoinnin opetteluun saattaa löytyä ohjelmistokehityksen eri osa-alueista, kuten esimerkiksi graafisesta suunnittelusta. Kaikista ei valmistu ohjelmoinnin ammattilaisia, mutta toimialojen digitalisoitumisen myötä monen työtehtävät saattavat tulevaisuudessa tavalla tai toisella liittyä ohjelmistokehitykseen (Ribbonfarm Consulting 2018). Laaja-alainen ymmärrys ohjelmistokehityksen eri vaiheista, rooleista ja tehtävistä edesauttaa työllistymistä, sillä moniosaajia tarvitaan entistä enemmän (Ängeslevä 28.8.2018). Järvenpää (2018) mainitsee kirjassaan, että kun hallitsee yhden ohjelmointikielen hyvin, oppii helposti muitakin ohjelmointikieliä.

Teollisen aikakauden aikana ohjelmia ei kirjoitettu tavallisten ihmisten, vaan teollisuuden tarpeisiin. Ohjelmoinnin teettäminen oli kallista ja hankalaa. Ohjelmia kirjoitettiin lähinnä laitteiden ohjaamiseksi. Ohjelmilla laskettiin asioita, joita ihminen ei pystynyt laskemaan tai joiden laskeminen olisi vienyt liian kauan aikaa. Tällaisia järjestelmiä ovat esimerkiksi pankkijärjestelmät, tietoverkot ja kassajärjestelmät. Nykyään digitaalisuuden aikakauden aikana viihdeteollisuus ja liiketoiminnat yleisesti on digitalisoitu. (Ribbonfarm Consulting 2018.) Tulevaisuuden ohjelmoijat tulevat koodaamaan ensisijaisesti ihmisten käyttämiä palveluita, kuten Facebook, Instagram, Whatsapp, Snapchat, Airbnb ja Uber. Aloittamisen helppouden ja motivaation kannalta on kuitenkin hyvä lähteä liikkeelle toteutuksista, jonka oppija ymmärtää, jota hän pääsee heti ohjelmoimaan ja jossa hän näkee heti työnsä tuloksen (Koodi 2016, 80).

2 Graafinen ohjelmointiympäristö Scratch

Scratch.mit.edu on ilmainen verkkopalvelu, joka tarjoaa käyttäjille ohjelmointieditorin, jolla Scratch-ohjelmia ohjelmoidaan. Verkkopalvelussa on valtava määrä ohjeita ja vinkkejä, valmiita muokattavissa ja kopioitavissa olevia ohjelmia, esimerkkejä, opasteita sekä wiki, jotka tukevat ohjelmointia ja ohjelmoinnin opettelua. Käyttäjät ovat kääntäneet Scratch-ohjelmointiympäristön käyttöliittymää yli 70 kielelle, myös suomeksi. (Scratch 2018.)

2.1 Legot Scratchin inspiraationa

Vuonna 1949 tanskalainen lelualan yritys nimeltä Lego alkoi valmistamaan muovisia, standardikokoisia palikoita. Ideana oli tarjota rajattomat mahdollisuudet rakentamiselle yhdistelemällä palikoita toisiinsa monin eri tavoin. Rakennelmia voi purkaa, jatkaa ja yhdistellä loputtomiin. (Lego Group 2018.) Legon idea standardoiduista, kierrätettävistä ja monistettavista rakennusosasista toistuu Scratch-ohjelmointiympäristössä (Reimer 23.7.2007).

Digitaalisuus antaa loputtomat mahdollisuudet rakentamiselle ja luovuudelle. Fyysiset rakennuspalikat saattavat loppua kesken, mutta digitaalisessa ympäristössä tällaista vaaraa ei ole. Ohjelmoinnissa ongelmaksi voikin koitua inspiraation ja ideoiden puute seuraavaan ohjelmoitavaan ohjelmaan. Siksi inspiraation ja uusien ideoiden löytämiseksi Scratch kannustaakin käyttäjiä jakamaan muille omaa koodia ja ohjelmia sekä muokkaamaan toisten kirjoittamia ohjelmia ja tekemään ohjelmista yhdistelmiä. Scratchin avulla ohjelmoinnin opetus voidaan järjestää yhtä helpoksi kuin legojen rakentaminen. Scratch kannustaa Legon tavoin mielikuvituksellisiin leikkeihin, jolloin luovat ideat syntyvät leikin avulla. Motivaation kannalta on hyvä lähteä liikkeelle toteutuksista, jotka aloitteleva ohjelmoija ymmärtää. Aloittelijan tulisi päästä heti ohjelmoimaan ja nähdä nopeasti työnsä tuloksen. (Liukas & Mykkänen 2014, 80.) Toimivan kokonaisuuden rakentaminen opettaa oppilaille ohjelmoinnin käsitteistöä ja algoritmista ajattelua (Peda).

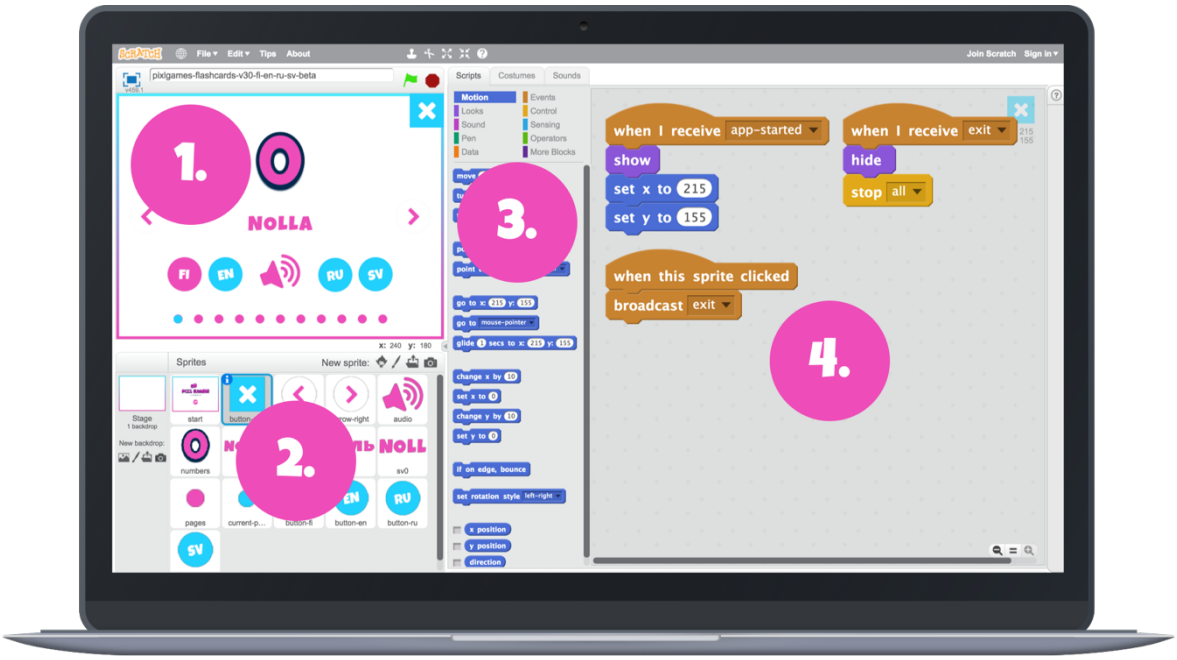
Scratch on onnistunut tekemään digitaalisessa muodossa sen, minkä Legot ovat tehneet fyysisinä rakennuspalikoina. Legon rakennussarjat toimitetaan paketteina, jotka sisältävät yksinkertaisen selkeät rakennuspalikat ja rakennusohjeet. Ohjeiden noudattaminen ei vaadi rakentajalta tekstin lukutaitoa, sillä ohjeet ovat visuaaliset ja yksityiskohtaisesti kuvitetut. Ohjeet etenevät palikka palikalta rakennelman alusta loppuun asti. Lego-palikat esitellään ohjesivuilla rakennusjärjestyksessä, minkä jälkeen ohjeistetaan, mihin kohtaan rakennelmaa ne sijoitetaan. Scratchin avulla ohjelmointi, ohjelmointiin tutustuminen ja ohjelmoinnin opetus voidaan järjestää yhtä helpoksi.

Visuaalinen ohjelmointiympäristö luo puitteet, jossa koodialueelle rakennetaan digitaalisista rakennuspalikoista ohjelman lähdekoodi. Lähdekoodi rakennetaan visuaalisten Skriptit-ryhmän lohkoista eli lauseista, jotka toimivat ohjelman komentopalikoina, toimintakäskyinä ja -ohjeina (Järvenpää 2018). Visuaalinen lähdekoodi ja Scratchin toimintaohjeet voidaan dokumentoida Legon tapaan yksityiskohtaisesti kuvitetuiksi vaihe vaiheelta eteneviksi ja selkeiksi ohjeiksi. Muita yhtäläisyyksiä Legon ja Scratchin välillä ovat hahmot ja taustat. Scratchin hahmot edustavat legoukkeleita. Scratchin taustat ovat kuin Legojen rakennusalustat. Scratchin näyttämöruutu edustaa toimintaohjeiden mukaan laadittua Legorakennelmaa. Scriptit, asusteet ja äänet ovat kuin mitä tahansa Lego-palikoita.

2.2 Scratch-ohjelmointiympäristön tärkeimmät osat inspiraationa

Scratch-ohjelmointieditorin voi asentaa tietokoneelle tai sitä voi käyttää suoraan modernissa internetselaimessa (Scratch 2018). Helppokäyttöinen editori on jaettu neljään eri osaan (Kuva 1):

- Stage eli näyttämö tai interaktiivinen ruutu, jossa ohjelmaa ajetaan ja ohjataan. Näyttämö vastaanottaa eri syötteitä, kuten esimerkiksi hiiren klikkaukset.
- Sprite list, jossa hallinnoidaan ohjelman graafisia elementtejä, esimerkiksi hahmoja ja taustakuvia.
- Blocks palette -osio, joka sisältää värikoodatut ohjelmointiskriptit, graafisten elementtien eri ilmentymät ja äänet.
- Scripts area, koodi alue, johon ohjelman visuaalinen lähdekoodi kootaan komentopalikoista.



Kuva 1: Scratch-ohjelmointiympäristön neljä osa-aluetta 1. Stage 2. Sprite list 3. Blocks palette 4. Script area.

Scratch-ohjelmointi on hahmokohtaista, eli jokainen hahmo sisältää oman lähdekoodin, jolla hahmoa ohjelmoidaan (Järvenpää 2018). Scratchin englanninkielisessä versiossa puhutaan spriteistä, ei hahmoista. Spritelle ei ole varsinaista suomenkielistä käännöstä, mikä on saattanut vaikuttaa siihen, että käyttäjät ovat kääntäneet spriten hahmoksi. Scratch-ohjelmoinnissa sprite ei välttämättä ole hahmo, vaan se voi olla mikä tahansa ohjelmassa käytetty graafinen elementti, jolla on eri olomuotoja. Esimerkiksi painike, jolla on eri olomuotoja: ei-aktiivinen, aktiivinen, painettu alas. Perusperiaatteena on, että sprite-listasta valitaan jokin graafinen elementti. Graafiseen elementtiin voidaan luoda toiminnallisuuksia (esimerkiksi liikettä) skriptien avulla. Graafinen elementti voidaan esittää eri olomuodoissa, ja siihen voi liittää ääniä. Ohjelmaa ei tarvitse erikseen kääntää konekielelle tai asentaa, vaan se käynnistyy suoraan näyttämön ruudulle. Näyttämön avulla koodin toimivuutta voi testata nopeasti, eli tehdyn työn tuloksen näkee heti.

MIT Scratch Teamin (2018) julkaiseman tiedotteen mukaan uusi Scratch-versio 3.0 tuo mukanaan merkittäviä uudistuksia. Uudistuksilla on pyritty parantamaan ohjelmien kehittämisen aloittamisen helppoutta, omien ohjelmien jaettavuutta ja ohjelmien sekoittamista toisiinsa. Yksi iso kehitysaskel Scratch-ohjelmointiympäristölle on tablet-laitteiden tuki. Scratch 3.0 -versiopäivityksen yhteydessä julkaistaan uusi järjestelmä laajennuksille, joka mahdollistaa fyysisten laitteiden (kuten micro:bit ja littleBits) ohjelmoimisen, sekä useiden verkkopalveluiden (kuten Google Translate) ohjelmointirajapintojen hyödyntämisen Scratch-ohjelmoinnissa. (MIT Scratch Team 2018.)

2.3 Scratchin käyttö opetuksessa

Ohjelmoinnillisen ajattelun ja algoritmisen ajattelun oppiminen on asetettu yhdeksi vuoden 2014 perusopetuksen opetussuunnitelman perusteiden päätavoitteeksi. Tämä on perusteltua muun muassa siksi, että ohjelmoinnin opetus on ennen kaikkea ajattelun opettamista (Liukas & Mykkänen 2014, 76–77). Algoritminen ajattelu ei ole välttämättä sidoksissa tietokoneisiin ja tietokoneohjelmiin, vaan algoritmista ajattelua voidaan opetella arkielämän ongelmilla. Tällöin ongelma pyritään purkamaan mahdollisimman pieniin osiin, joita lähdetään ratkaisemaan yksi kerrallaan järjestyksessä (Liukas & Mykkänen 2014, 77). Ongelmien ratkaisemista helpottaa, jos osaa tunnistaa niissä kaavoja ja säännönmukaisuuksia. Arkielämän ongelmien ratkaisussa auttava kaavojen ja säännönmukaisuuksien tunnistaminen on hyödyllinen kyky myös ohjelmoinnissa. Liukkaan & Mykkäsen (2014, 78) mukaan ohjelmoinnissa algoritmejä voidaan verrata kokkikirjan aterian reseptеihin ja valmistusohjeisiin.

Scratchissä ohjelmointi tapahtuu visuaalisesti. Visuaalisuus helpottaa koodin lukemista ja ohjelman rakenteen hahmottamista. Scratch soveltuu visuaalisuutensa vuoksi opetukseen, sillä koodia ei varsinaisesti kirjoiteta. Näin vältytään kirjoitusvirheiltä. Ohjelmakoodi rakennetaan värikoodatuista ja yksilöllisesti muotoilluista komentopalikoista. Aloittelijan ei välttämättä tarvitse osata lukea tai kirjoittaa aloittaakseen Scratch-ohjelmoinnin. Visuaalisessa ohjelmointiympäristössä ongelmakohtien paikantaminen ja havaitseminen on helpompaa. Oppilas voi kopioida lähdekoodin ensin omaan ohjelmaansa, minkä jälkeen hän voi muokata sitä luovasti.

Ohjelmoinnin oppimisen kannalta on tärkeää, että harjoitteet on liitetty oppilaiden arkeen tai muihin oppiaineisiin ja niille asetetut opetukselliset tavoitteet koetaan merkityksellisiksi. Oppimiseen vaikuttavat tehtävät, joiden työstäminen koetaan mielekkääksi. Scratch-ohjelmointi koetaan mielekkääksi sen moninaisuuden ansiosta. Scratchillä voi ohjelmoida pelejä, animaatioita, kertomuksia, ja sillä voi tehdä erilaisia simulaatioita. (Neuman 2017, 21.) Scratch mahdollistaa aloittelijalle hyvän ensikokemuksen ohjelmoinnista ja saa hänet näkemään nopeasti työn tulokset. Tämä motivoi jatkamaan ohjelmointikielen tutkimista ja opiskelua. (Liukas & Mykkänen 2014, 100.)

2.4 Scratch taipuu moneen

Neuman (2017, tiivistelmä) on gradussaan tutkinut, onko ohjelmointi perusopetuksen opetussuunnitelman perusteiden sisältämän tavoitteen mukaisesti innostavaa. Hän on toteuttanut tutkimuksen haastattelemalla yhden kuudennen luokan oppilaita ja opettajaa. Haastatellulla opettajalla ei ollut aikaisempaa ohjelmointikokemusta, eikä hänellä ollut

kokemusta muista ohjelmointiympäristöistä kuin Scratchista. Opettajan mukaan Scratchin tarjoamista toimintavihjeistä ja käyttöoppaasta oli opetuksen aloittamisessa apua (Neuman 2017, 37). Osa Neumanin haastattelemista lapsista koki varsinaista opetusta olevan liian vähän ja itsenäistä opiskelua liian paljon. Tämä katsottiin huonoksi asiaksi, sillä itsenäinen opiskelu ilman tukea vei huomattavasti enemmän aikaa. Osa lapsista koki, ettei koulussa opeteta tarpeeksi vaikeita asioita. Oppilaat pitivät Scratchilla ohjelmointia helppona. Toisaalta he kokivat, että Scratchilla ei voi tehdä kuin yksinkertaisia ja helppoja juttuja. (Neuman 2017, 41.)

Scratchilla voidaan tehdä hyvin monimutkaisia ohjelmia, kuten erilaisia todellisuutta mallintavia simulaatioita. Todellisuutta havainnollistavia, muokattavia Scratch-projekteja on saatavilla useasta eri kategoriasta esimerkiksi elämä, lentäminen, matematiikka, keskustelu-botit ja fysiikka. Fysiikkasimulaatiolla voi jäljitellä esimerkiksi painovoimaa tai esineiden törmäysten vaikutusta. Scratchilla on tehty myös käyttöjärjestelmiä ja laitteita jäljitteleviä projekteja. Elämää voi simuloida esimerkiksi suosituilla virtuaalilemmikeillä. Scratchilla on ohjelmoitu esimerkiksi lentosimulaattori, joka mahdollistaa lentämisen virtuaalisesti. (Scratch Wiki 2018.) Valmiita ja kaikkien saataville julkaistuja projekteja löytyy Scratch-verkkopalvelusta tuhansia. Projekteja on saatavilla eri kategorioiden alta, kuten animaatiot, pelit, tarinat ja musiikki (Scratch 2018).

3 Ohjelmistokehityksen oppimateriaalin tuottaminen

Suomen uudessa, alakoulun osalta vuonna 2016 käyttöönotetussa, opetussuunnitelmassa ohjelmointi on kytketty matematiikan oppiaineeseen. Tästä syystä ohjelmointiin kulutettava aika varataan matematiikan tuntijaosta. Suomessa koulut ja opettajat päättävät, miten paljon aikaa ohjelmointiin käytetään, sillä kiinteää tuntimäärää ei ole annettu. (Liukas & Mykkänen 2014, 70.) Monesta muusta Euroopan maasta poiketen Suomessa ohjelmointia ei tuotu opetussuunnitelmaan täysin omana kokonaisuutena, jolla olisi omat erilliset tavoitteet muista aineista (Raivonen 2018). Tämä on Hjelmin (2016) mukaan saattanut johtaa siihen, että ohjelmointia ei ymmärretä tai osata soveltaa luovuuden välineenä.

Viro ja Iso-Britannia ovat ensimmäisinä Euroopassa ottaneet ohjelmoinnin opetussuunnitelmaansa. Viro aloitti järjestelmällisen ohjelmoinnin opetuksen vuonna 2013 ja Iso-Britannia vuonna 2014. (Liukas & Mykkänen 2014, 62.) Ohjelmoinnin opetus ollaan ottamassa osaksi alakoulun opetussuunnitelmaa monissa Euroopan maissa. Ohjelmoinnin pääpaino on näissä maissa matemaattisten ajattelutaitojen opetuksessa. Ohjelmoinnin opetus on pääasiassa yhdistetty muihin oppiaineisiin (Raivonen 2018). Iso-Britannia opettaa ohjelmointia omana Computing-nimisenä oppiaineenaan (Liukas & Mykkänen 2014, 64). Haaste on kuitenkin kaikkialla sama – pedagogisen ymmärryksen puute alakouluikäisten ohjelmoinnin opetuksessa. Ei tiedetä, pitäisikö aloittaa tekemisellä vai teorialla. Ikäkaudelle sopivan ohjelmointikielen valinta tuottaa haasteita, varsinkin jos aikaisempaa kokemusta ohjelmoinnista, ohjelmoinnin opetuksesta ja ohjelmoinnin työvälineistä ei ole. (Liukas & Mykkänen 2014, 62.)

3.1 Innostuminen ja harjoittelu opettelun ytimessä

Ohjelmoinnin opetuksessa tärkeintä on saada nuoret aloittelevat ohjelmoijat ymmärtämään algoritmista ajattelua ja sen soveltamista ohjelmoinnissa. Alakoulussa ohjelmoinnin opettaminen on vielä kaukana varsinaisesta monimutkaisten ohjelmien teosta. Tärkeää on oppia, millaisia asioita tietokone on hyvä tekemään ja miten tietokoneelle annetaan yksikäsitteisiä ohjeita. (Liukas & Mykkänen 2014, 68.) Opettajan ei varsinaisesti tarvitse osata ohjelmoida. Tärkeämpää on, että hän osaa selittää, miksi ohjelmoinnin osaaminen on tärkeää ja osaa opastaa oppilaita harjoitustehtävissä ja niihin liittyvissä haasteissa. (Liukas & Mykkänen 2014, 69.) Oppimisen kannalta on tärkeää saada oppilaat innostumaan ohjelmoinnista. Tästä syystä on tärkeätä olla näyttämättä oppilaille innottomuutta, jos opettaja jostain syystä ei itse ole innostunut ohjelmoinnista (Liukas & Mykkänen 2014, 69).

Liukas & Mykkänen (2014, 80) kirjoittavat, että ohjelmoimaan oppii vain itse koodaamalla. Taustalla olevan teorian voi oppia kuka tahansa, mutta todellinen osaaminen tulee vain oman yrityksen ja erehdyksen kautta. Heidän (Liukas & Mykkänen 2014, 80) mukaansa ohjelmointikyvyt ovat verrannollisia siihen, minkä verran on ohjelmoinut, miten monta tuntia ja millaisia ongelmia on kohdannut ja miten etsinyt niihin parasta ratkaisua. Tästä syystä Liukas & Mykkänen (2014, 80) eivät koe, että ohjelmoinnin opetuksen tulisi olla teorian tankkaamista, vaan heidän mukaansa opetuksen yhteydessä oppilaiden olisi päästävä heti tekemään ja kokemaan oman työnsä tulokset näytöllä. Tästä syystä tämän opinnäytetyön toiminnallinen osa keskittyy tekemiseen eikä ohjelmoinnin teoriaan.

3.2 Oppimateriaalikokonaisuun suunnittelun lähtökohdat

Opetushallitus päättää perusopetuksen opetussuunnitelman perusteet, jonka pohjalta opetuksen järjestäjä (yleensä kunta tai koulu) laatii paikallisen opetussuunnitelman. Paikallinen opetussuunnitelma määrittelee opetusuunnitelman perusteita tarkemmin, miten kouluissa järjestetään opetusta. Opetussuunnitelmassa määritellään opetuksen tavoitteet ja sisällöt. Vuoden 2014 perusopetuksen opetussuunnitelman perusteisiin otettiin uudeksi sisällöksi ohjelmoinnin opetus alaluokilta lähtien. (Opetushallitus 2014.) Ohjelmoinnin opettaminen on herättänyt keskustelua. Esimerkiksi OAJ on kommentoinut, että opettajien perehdytys uuteen opetussuunnitelmaan on riittämätöntä (Mansikka 2016).

Ohjelmoinnin opettaminen ja siihen liittyvät oppimateriaalit ovat kiinnostaneet minua ja olen aktiivisesti seurannut niihin liittyvää keskustelua esimerkiksi Facebook-ryhmässä “Tieto- ja viestintätekniikka opetusessa/ICT in Education”. Halusin tarjota mallin ohjelmistokehityksen oppimateriaalin laatimiseksi vuosiluokille 3–6. Tämän opinnäytetyön toiminnallinen osa muodostaa oppimateriaalikokonaisuuden, jonka tavoite on innostaa oppilaita laatimaan tietokoneohjelmia graafisessa ohjelmointiympäristössä. Kokonaisuus vastaa opetussuunnitelman matematiikan tavoitteeseen suunnitella ja toteuttaa tietokoneohjelma graafisessa ohjelmointiympäristössä. (Opetushallitus 2014.)

Tämän opinnäytetyön materiaalikokonaisuus ratkaisee yhtä isoa ongelmaa jakamalla sen pienemmiksi osiksi. Tarkoituksena on havainnollistaa ohjelmoinnilla ratkaistavien haasteiden moninaisuutta ja tukea algoritmisen ajattelun opettelua. Materiaalin suunnittelussa on pyritty ottamaan huomioon oppilaiden tarve luoda ohjelmista persoonallisia. Ohjelmointitehtävän ratkaiseminen tuntuu oppilaasta mielekkäämmältä, kun oppilas voi määrittää ohjelman värit, kuvat ja äänet itse (Neuman 2017). Virheiden korjaamisen kannalta on kuitenkin järkevää, että materiaalikokonaisuuden avulla muodostuva ohjelma rakennetaan ensin ohjeiden mukaisesti, ennen kuin siitä lähdetään rakentamaan luovia versioita.

Laajan kokonaisuuden on tarkoitus poistaa harhakäsityksiä siitä, että Scratch ei olisi oikea ohjelmointikieli tai että sillä voidaan tehdä vain hyvin yksinkertaisia ohjelmia.

Oppimateriaalikokonaisuus on suunniteltu jaettavaksi ja käytettäväksi täysin digitaalisesti eri laitteilla ja eri kokoisilla ruuduilla (Kuva 2). Kokonaisuus muodostuu graafisen ohjelmointiympäristön eli Scratch-ohjelman SB2-tiedostosta ja siihen liittyvästä ohjelmistokehitykseen perehdyttävästä tehtävävihkosta. SB2-tiedosto sisältää täysin toimivan Scratch-ohjelmointikielellä ohjelmoidun ohjelman, jonka graafinen lähdekoodi on purettu osiin. Osiksi purettu ohjelma kootaan toimivaksi tehtävävihkon vaihe vaiheelta etenevien yksinkertaisten ohjeiden avulla. Opetussuunnitelmassa ohjelmoinnin opetuksen tavoitteena vuosiluokilla 3–6 on graafiseen ohjelmointiympäristöön tutustuminen ja ohjelmoinnin peruskäsitteiden oppiminen. Vaiheittain etenevä tehtävävihko antaa oppilaille tähän mahdollisuuden, ja he pääsevät aloittamaan ohjelmoinnin ilman aikaisempaa ohjelmointikokemusta. Digitaalinen tehtävävihko on suunniteltu siten, että sitä on helppo lukea puhelimen, tabletin tai tietokoneen ruudulta. Osa tehtävävihkon tehtävistä on suunniteltu tehtäväksi kynällä ja paperilla, joten vihko tai osa sen sivuista voidaan tarvittaessa tulostaa paperille.

Kuva 2. Tämän opinnäytetyön toiminnallinen osa, Scratch-ohjelma ja tehtäväkirja eri laitteissa

Perusopetuksen opetussuunnitelman perusteiden mukaan opetuksessa on otettava huomioon oppilaiden erilaiset etenemistahdit ja tavat oppia. Oppilaille on annettava mahdollisuus edetä yksilöllisesti (Opetushallitus 2014). Tämän opinnäytetyön vaiheistettu kokonaisuus on toteutettu siten, että oppilailla on mahdollisuus yksilölliseen etenemiseen.

Ylöspäin eriyttävää sisältöä tässä ohjelmistokehityksen oppimateriaalikokonaisuudessa ovat esimerkiksi animoinnit ja sivunumerot.

Ohjelmointia oppii parhaiten ohjelmoimalla itse (Järvenpää 2018). Oppilaan on tarkoitus oppia käyttämään erilaisia ohjelmistoja ja palveluita tehtäväkirjan tehtäviä suorittaessaan. Samalla oppilas saa harjoitusta tekstin tuottamisesta ja käsittelystä eri välineillä. Tehtäviä tekemällä oppilas saa kokemusta kuvan, äänen ja animaatioiden tekemisestä. Tehtävävihkon tehtävät on suunniteltu siten, että oppilas voi tehdä ne joko yksin tai ryhmässä toisten kanssa. Ohjelmointi on tuotu työelämän kontekstiin ja sitä käsitellään kurssin aikana yhtenä työvaiheena ohjelmistokehitystä. Näin oppilas saa kokemusta siitä, kuinka teknologian toiminta riippuu ihmisten tekemistä ratkaisuista. Nämä taidot on kirjoitettu osaksi uuden opintosuunnitelman tieto- ja viestintäteknologisen laaja-alaisen osaamisen tavoitteita vuosiluokilla 3–6. (Opetushallitus 2014.)

Oppimateriaalin suunnittelussa ja toteutuksessa on pyritty huomioimaan projektien toteuttamisen harjoittelu sekä ryhmässä toimiminen. Oppimateriaaliin on sisällytetty toimintaa, jossa oppilaat saavat kokemusta ohjelmistokehityksestä työelämässä ja siinä tarvittavista tehtävistä ja rooleista. Nämä taidot liittyvät uuden opintosuunnitelman työelämätaitojen osaamistavoitteisiin vuosiluokilla 3–6. (Opetushallitus 2014.)

Halmeen (2018) mukaan ohjelmistokehityksestä puhuttaessa ja kirjoittaessa on usein turvauduttava vertauskuviin, kuten talonrakentamiseen tai esimerkiksi Lego-palikoilla rakentamiseen. Helposti ymmärrettävät vertauskuvat havainnollistavat ohjelmoinnin moninaisuutta. Tässä opinnäytetyössä tuon ohjelmistokehityksen Lego-metaforan konkreettiselle tasolle. Opinnäytetyön toiminnallisen osan tehtävävihko noudattaa Legon rakennussarjojen ideaa ja logiikkaa havainnollistettaessa ohjelmistokehityksen eri vaiheita. Tehtävävihko sisältää vaihe vaiheelta etenevät selkeät kuvitetut rakennusohjeet, joiden avulla ohjelmisto saadaan kehitettyä.

3.3 Oppimateriaalin ja tuotekehitysprosessin vaiheet

Opinnäytetyöni yhtenä tavoitteena oli suunnitella ja kehittää tuote, jota voisi käyttää ohjelmoinnin opetuksessa alaluokilla. Tämän tuotteen (eli opinnäytetyön toiminnallisen osan) muodostavat Scratch-ohjelma ja tehtävävihko. Tuotteen kaikki elementit ja osat ovat toteutettavissa ilmaisilla verkossa toimivilla palveluilla ja työkaluilla. Suurin osa palveluista on vapaasti käytettävissä ilman tunnuksia tai kirjautumista.

Käyttökelpoinen materiaali ohjelmistokehityksen opetukseen tuntuu puuttuvan tai ainakin uudenlaiselle materiaalille tuntui olevan tarvetta. Syntyi ajatus toteuttaa sellainen. Ensimmäinen idea oli lähteä jäljittelemään vanhojen tietokonelehtien mukana tulevia, tekstistä kopioitavia ohjelmia. Omassa lapsuudessani kotitietokoneet yleistyivät. Kokoonnuimme kyläkouluni tietokoneluokkaan, jossa meille opetettiin tietokoneiden käyttöä pelien muodossa. Jotta pelejä päästiin pelaamaan, ne jouduttiin usein ohjelmoimaan ensin. Tämä tapahtui kopioimalla lähdekoodi lehtien ja kirjojen sivuilta. Ystäväpiirissäni tämä tapa tutustua ohjelmointiin on toiminut monelle ponnahduslautana ammattimaiseen ohjelmistokehitykseen. Nykyään sama ohjelmoinnin opettamisen tapa voidaan toteuttaa Scratch-ohjelmointiympäristössä.

Valmiin ja visuaalisen lähdekoodin kopiointiin soveltui Legon malli. Syntyi konsepti, josta rakensin prototyypin Scratch-ohjelmointiympäristössä. Prototyypin ympärille kehitin tuotteen visuaalisen ilmeen ja identiteetin, joka muodostuu Pixl Games -nimestä, logosta, väreistä ja fonteista. Yksi merkittävä osa prosessia oli jatkuvan kehityksen malli, jossa jokainen prosessin eri vaihe toteutettiin ”rakenna - mittaa - opi” -periaatteella, jota toistetaan parhaan lopputuloksen saavuttamiseksi.

Opinnäytetyön toiminnallisen osan tuotantovaiheet:

1. Konseptisuunnittelu, johon sisältyi kohderyhmään tutustuminen, joka toteutettiin haastattelemalla eri opettajia ja kustannustoimittajia, jotta tarve saatiin määriteltyä
2. Tekninen suunnittelu, käyttöliittymäsuunnittelu ja sisältösuunnittelu
3. Graafinen suunnittelu, jonka lopputuloksena syntyi sekä ohjelman että tehtävävihkon visuaalinen ulkoasu
4. Ohjelmointi

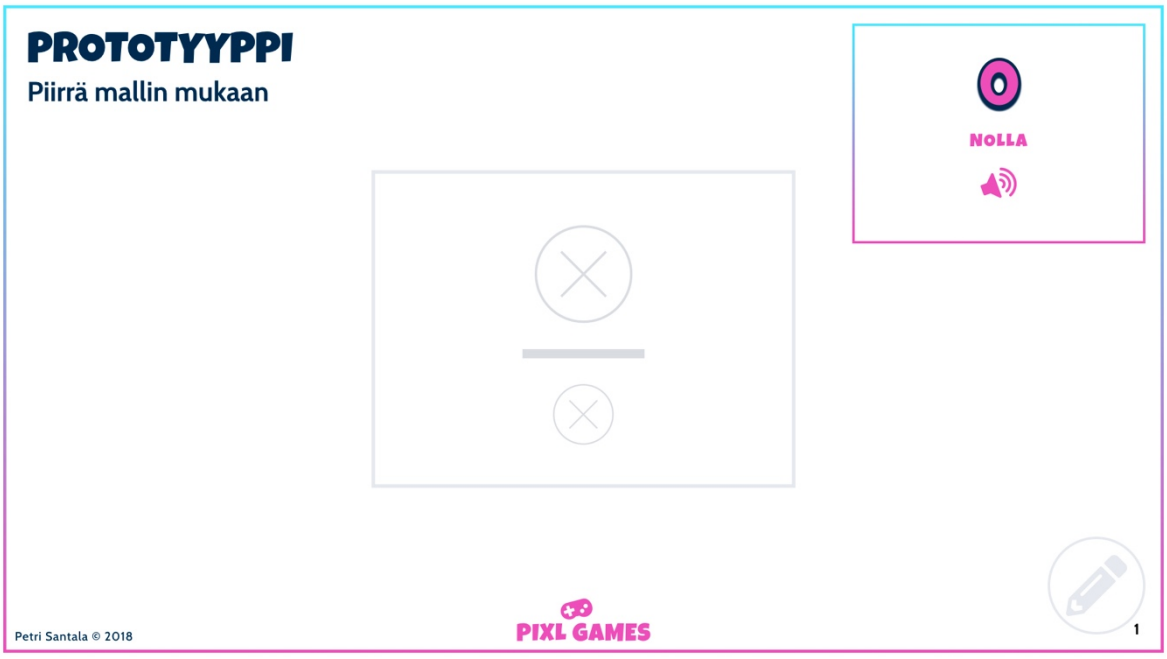
3.4 Ohjelmistosuunnittelun ja -kehityksen roolit, tehtävät ja työkalut

Ohjelmistokehitys on muuttunut siten, että ohjelmia ja pelejä kehitetään tiimeissä, jotka muodostuvat moninaisista osaajista. Tärkeimmässä roolissa eivät ole enää ohjelmoijat. Kun tiimiläiset ymmärtävät toistensa roolit ja tehtävät, on koko tiimin helpompi hahmottaa kokonaisuus ja se mitä ollaan tekemässä. (Ängeslevä 28.8.2018.)

Ohjelmistokehitys voidaan jakaa karkeasti muutamaan eri rooliin ja tehtävään, joita ovat määrittely, suunnittelu, ohjelmointi, projektijohto (Rose 2017). Tämän opinnäytetyön toiminnallinen osa muodostaa projektin, jonka projektipäällikkönä toimii opettaja. Opettaja toimittaa oppilaille asiakkaan laatiman määrittelydokumentin. Oppilaiden tehtävä on suunnitella ja toteuttaa tietokoneohjelma määrittelydokumentin pohjalta opettajan johdolla.

Tehtäväkirjassa ohjelmistokehitys on jaettu eri vaiheisiin: käyttöliittymäsuunnittelu, graafinen suunnittelu, tekstisuunnittelu, äänituotanto ja tekninen toteutus eli ohjelmointi. Vaiheet on pilkottu osiin ja paketoitu pieniksi kokonaisuuksiksi, jotka on helppo toteuttaa yksilö- tai ryhmätyönä. Jokaisessa vaiheessa hyödynnetään ohjelmistokehityksessä tarvittavia taitoja, kuten graafinen suunnittelu ja ohjelmointi.

Käyttöliittymäsuunnittelussa määritellään ohjelman tarpeelliset toiminnot. Ohjelman tarpeelliset elementit pyritään suunnittelemaan niin, että valmis tuote on käyttäjälle selkeä ja yksinkertainen käyttää. (Muranen & Harmainen 2017.) Käyttöliittymäsuunnittelua käsitellään tehtäväkirjassa siten, että oppilaat voivat piirtää kynän ja paperin avulla oman käyttöliittymän ohjelmalle. Käyttöliittymä piirretään paperille yksinkertaisina laatikoina ja palkkeina. Piirretyt laatikot edustavat jotakin ohjelman elementtiä, jonka toiminnallisuus on kuvattu erikseen. Oppilas voi piirtää elementit ruudulle haluamallaan tavalla. Jotta kokonaisuus pysyy hallittavana, ohjeistetaan oppilaita laatimaan lopullinen käyttöliittymä mallin mukaisesti.



Kuva 3. Tämän opinnäytetyön tehtävävihkon esimerkki käyttöliittymäsuunnittelusta

Tehtäväkirjan graafisessa suunnitteluvaiheessa oppilas suunnittelee, miltä ohjelman eri elementit näyttävät (Kuva 3). Oppilas hyödyntää luovuuttaan miettiessään, minkä muotoisina ja värisinä paperille laatikkoina hahmotellut ohjelman eri elementit lopulta kuvitetaan. Graafisen suunnittelun työkaluina voidaan käyttää paperia ja värikyniä tai eri laitteissa valmiiksi olevilla piirto-ohjelmilla. Jos laitteessa ei ole valmista ohjelmaa, voi käyttää

esimerkiksi ilmaista selaimessa toimivaa SumoPaint-kuvankäsittelyohjelmaa (SumoPaint). Oppilaat suunnittelevat ohjelman ulkoasusta oman version. Varsinainen toteutus tehdään valmiilla elementeillä, jotta opettajan on helpompi seurata tehtävän etenemistä. Virheiden tarkistus ja oppilaan ohjaus on helpompaa, kun kaikki tekevät samaa ohjelmaa samoilla rakennuspalikoilla.

Ohjelmassa ja tehtäväkirjassa käytetyt fontit on hankittu Google Fonts -palvelusta, joka tarjoaa laajan kirjon erilaisia ilmaisia fontteja eri käyttötarkoituksiin. Kaikki Google Fonts -palvelun fontit ovat kaikille ilmaisia ja kaikkien saatavilla (Google Fonts).

Oppimateriaaleissa käytetyissä symboleissa ja ikoneissa on käytetty Material Design ja Font Awesome -palveluiden ilmaisia fontti-ikoneita. Sekä Material Design että Font Awesome sisältävät kumpikin reilut 900 erilaista ilmaista ikonia tai symbolia kaikenlaiseen, myös kaupalliseen käyttöön. (Font Awesome; Material Design.)

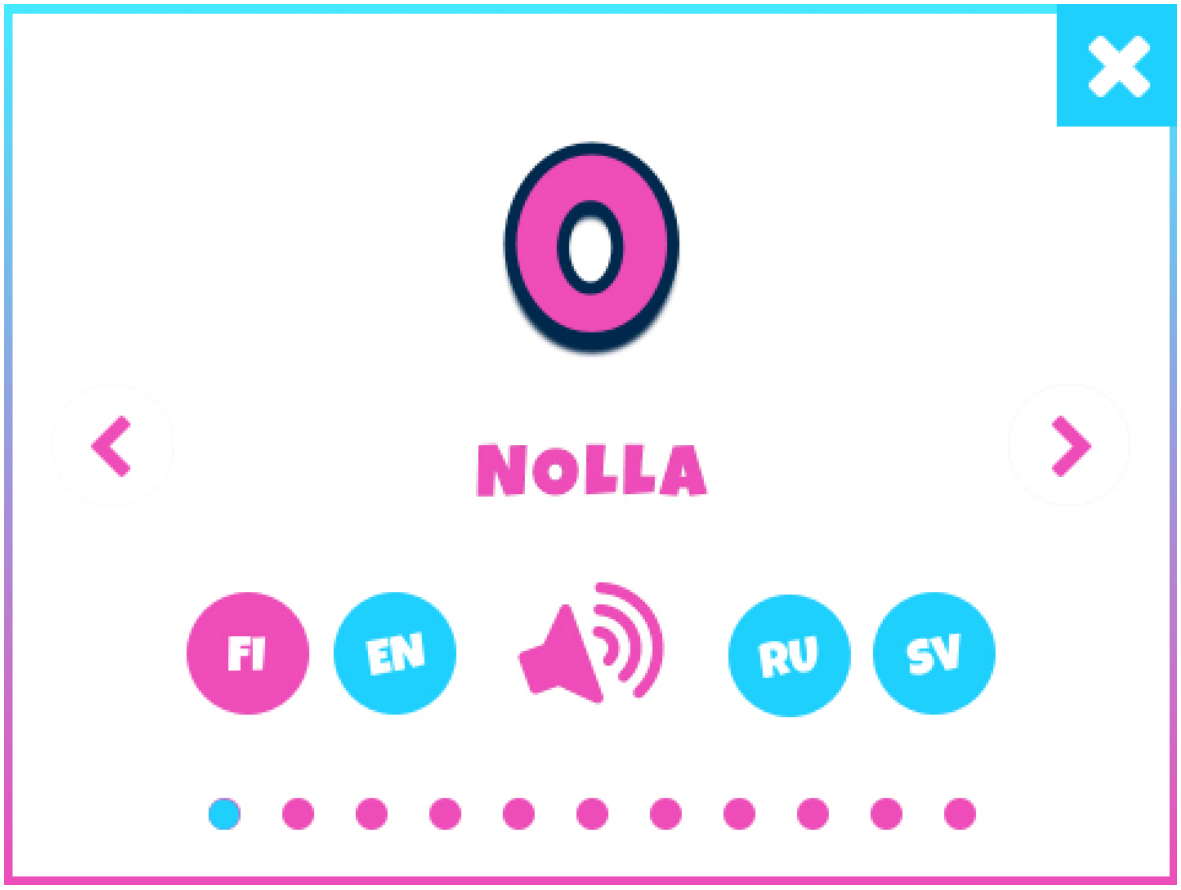
Tehtäväkirjassa toteutettavan kieltenopiskeluohjelmiston yksi keskeisimpiä elementtejä ovat numerot ja miten ne kirjoitetaan. Numerot on valittu aiheeksi, koska ne tunnetaan yleisesti kaikkialla maailmassa, niitä voi myös keksiä lisää tai niistä voi tehdä yhdistelmiä. Numerot ja luvut on helpompi kääntää eri maiden kielille, kun käännettävä asia on ennestään tuttu. Oppilaan ei tarvitse käyttää energiaa ymmärtääkseen, mitä ohjelmassa esitettävissä kuvissa on tai miten kuvat liittyvät toisiinsa. Halutessaan oppilas voi vaihtaa aihealuetta ja tehdä ohjelmasta oman version, vaikka metsän eläimillä tai eri maiden lipuilla. Ohjelman personointi voi tehdä ohjelmoinnista mielekkäämpää ja näin innostaa oppilasta oppimaan ohjelmointia (Neuman 2017, 21).

Tekstisuunnittelijan roolissa oppilas laatii ohjelman tekstit eli numerot merkkeinä ja auki kirjoitettuna. Auki kirjoitetut tekstit käännetään eri kielille. Käännös voidaan toteuttaa esimerkiksi maksuttoman Google Translate -palvelun avulla, jolla voi kääntää sanoja suomen ja yli sadan kielen välillä (Google Translate). Jos oppilailla ei ole mahdollisuutta käyttää tietokonetta tai sanakirjaa, voi käännökset kopioida suoraan tehtäväkirjasta. Tehtäväkirjan ja ohjelman vieraskieliset tekstit on käännetty Google Translate -verkkopalvelun avulla. Käännöstyön voi toteuttaa fyysisten tai verkossa toimivien sanakirjojen avulla. Wikipediaa voi käyttää luovasti yksittäisten sanojen kääntämiseen eri kielille.

Äänisuunnittelijan rooli ohjelmistokehityksessä on suunnitella ja toteuttaa äänet ohjelman tarpeisiin (Faller 2.6.2017). Ohjelmassa tarvittavat äänet voidaan toteuttaa ilmaisen Sound of Text -palvelun avulla. Sound of Text -palvelu muuttaa tekstin puheeksi ladattavina ja tietokoneella toistettavissa olevina MP3-äänitiedostoina. Palvelu käyttää Google Translate

-palvelun puhesynteesiä hyödyksi tekstin muodostamiseksi äänitiedostoksi eri kielille. Palvelua ylläpitävä Nick Pierson kehitti Sound of Text -verkkopalvelun alun perin omiin tarpeisiinsa. Palvelua käyttävät nykyään tuhannet ihmiset eri käyttötarkoituksiin. (Sound of text.) Oppilaat tuottavat jokaisen ohjelmassa käytettävän tekstin omaksi äänitiedostoksi. Kunkin kielen kohdalla käytetään kyseisen kielen ääntä. Ohjelmassa tarvittavat äänitiedostot löytyvät valmiiksi toteutettuina tiedostoina Scratch-ohjelmapaketissa.

Ohjelmoijan rooli on kirjoittaa yksityiskohtaiset toimintaohjeet tietokoneelle eli ohjelma jollakin ohjelmointikielellä (Järvenpää 2018, 10). Tehtävävihko opastaa oppilasta laatimaan tietokoneohjelman graafisessa ohjelmointiympäristössä ja ohjelmoimaan kaikki elementit yhdeksi toimivaksi kokonaisuudeksi (Kuva 4).



Kuva 4. Kuvakaappaus tämän opinnäytetyön toiminnallisen osan Scratch-ohjelmasta

Kehitettävän ohjelman testaus on tärkeä osa ohjelmistokehitystä (Halme 2018). Testaamisella ja laaduntarkastuksella pyritään takaamaan ohjelman toimivuus ja välttämään ohjelmointivirheet julkaistavassa ohjelmassa. Oikeaan aikaan ajoitettu virheiden ja toimivuuden tarkistus sekä löydettyjen ohjelmointivirheiden korjaaminen kehitysvaiheessa on edullisempaa, kuin julkaistun ohjelman korjaaminen jälkikäteen. (Derwin 2016.) Opinnäytetyön toiminnallisessa osassa testaus on sisäänkirjoitettu oppilaille laadittuihin tehtäviin.

Ohjelmoitavan sovelluksen jokainen elementti sisältää jonkin helposti havaittavan toiminnallisuuden. Toiminnallisuus voi olla ääni, animaatio tai kuvan vaihtuminen ruudulla. Jokainen toiminnallisuus käynnistetään ohjelmaa käyttävän käyttäjän toimesta. Toiminnallisuus voi käynnistyä esimerkiksi hiiren klikkauksesta tai kursorin viemisestä elementin päälle. Jos mitään ei tapahdu, voivat opettaja ja oppilas tarkistaa yhdessä visuaalisesti koodatut ohjelman toimintaohjeet vian löytämiseksi.

Tehtäväkirjan voi laatia Googlen ilmaisella selaimessa toimivalla Slides-ohjelmalla. Tällöin myös kaikki Google Fonts -palvelussa olevat fontit ovat käytössä ilman erillistä asennusta. Slides-ohjelman avulla voi laatia visuaalisia esityksiä, joihin voi tuoda kuvan lisäksi ääntä ja videota. Ohjelma on yhteensopiva Microsoftin vastaavan ohjelman, PowerPointin kanssa. Esitykset voi tallentaa PDF-muodossa, jolloin esityksen käyttäminen ei vaadi Googlen palvelun käyttöä. (Google Slides.)

4 Pohdinta

Opinnäytetyön aihe osoittautui työn edetessä henkilökohtaiseksi intohimon kohteeksi ja sydämen asiaksi. Mielenkiinto mielekkääseen aiheeseen auttoi toiminnallisen osan suunnittelussa ja toteutuksessa. Toiminnallisessa osassa hyödyntämääni Scratch-ohjelmointiympäristöön liittyy ennakkoluuloja ja sitä koskevassa keskustelussa olen huomannut, että Scratchin kaikkia mahdollisuuksia ei ymmärretä. Oma ennakkokäsitykseni oli alun perin saman suuntainen. Ajatus kuitenkin vaihtui ohjelmaa laatiessa ja Scratchin dokumentaatiota tutkiessa. Scratch-ohjelmointi osoittautui monipuoliseksi ja helppoutensa takia nopeaksi tavaksi laatia ohjelmia.

4.1 Opinnäytetyöprosessin kriittinen tarkastelu

Minulla on vahvoja mielipiteitä liittyen ohjelmointiin ja sen opettamiseen. Opinnäytetyötä kirjoittaessa oli haastavaa löytää tietoperustaa, joka tukisi mielipiteitäni. Tämä on saattanut johtaa siihen, että toiminnallinen osa voi osittain olla pedagogisesti puutteellinen. Pedagogiikkaa, aihealuetta tai ohjelmassa käytettyjen elementtien, värien tai fonttien mielekkyyttä ei ole validoitu kohderyhmän edustajilla. Ajan puutteen vuoksi toiminnallisen osan ohjelmointitehtävän vaativuutta tai laajuutta ei ehditty testaamaan kohderyhmää vastaavilla lapsilla tai ohjelmoinnista kiinnostuneilla henkilöillä.

Opinnäytetyötä kirjoittaessa törmäsin toiseenkin tietoperustaan liittyvään haasteeseen. Ohjelmistokehityksestä, ohjelmoinnista ja ohjelmoinnin opetuksesta on kirjoitettu verkossa paljon, mutta suurta osaa kirjoituksista ei ole toimitettu. Kirjoitustyön edetessä kirjoittaminen ja viittaaminen oli helpompaa silloin, kun tietoperustana oli kustannustoimitettu kirja tai jokin muu aineisto. Haasteen toi myös rajaamisen vaikeus, joka näkyy lopputuloksessa. Opinnäytetyön tehtävävihkoa olisi pitänyt tehdä samaan aikaan, kun koodasin toiminnallisen osan Scratch-ohjelmaa. Samanaikainen ohjelman dokumentointi olisi helpottanut hahmottamaan lopputuloksen laajuutta ja sitä kautta rajaamaan tiukemmin työtä. Pedagoginen lähestymistapa olisi voinut tehdä työstä mielenkiintoisemman.

Oppimateriaalikokonaisuuden laatiminen luokille 3–6 ohjelmistokehityksen näkökulmasta tuntui tärkeältä opinnäytetyötä tehdessä. Jälkikäteen se tuntuu kuitenkin liian laajalta kokonaisuudelta. Työssä olisi voinut keskittyä enemmän ohjelmaan, Scratch-ohjelmointiin ja tarinalliseen tehtävänantoon. Ohjelmistokehityksen prosessit ja roolit olisi voinut tuoda mukaan yläkoulun oppimateriaaleissa.

4.2 Kehittäminen ja jatkotoimenpiteet

Opinnäytetyön toiminnallinen osa perustuu moniin olettamuksiin, joita olisi syytä tutkia tieteellisin menetelmin. Mallista oppimisen mielekkyys ohjelmoinnin opetuksessa on yksi olettamus, joka perustuu omiin ja kollegoiden kokemuksiin. Aikomukseni on testata materiaalia. Viimeistelty tehtävävihko ja Scratch-ohjelma on tarkoitus julkaista osoitteessa www.pixlgames.fun, josta sen voi ladata opetus- tai tutkimuskäyttöön. Tehtävävihko julkaistaan avoimella Creative Commons versio 4.0 lisenssillä. Scratch-ohjelma julkaistaan avoimella MIT-lisenssillä. Scratch-ohjelmasta julkaistaan kaksi eri versiota: valmiiksi koodattu versio ja palasiksi purettu versio. Valmiiksi ohjelmoitu versio on tarkoitettu opettajille malliksi. Julkaisun jälkeen olen suunnitellut keräväni käyttäjäpalautetta jatkokehitystä varten. Palautteen pohjalta samalla idealla voisi toteuttaa oppimiskokonaisuuksia joissa ohjelmoinnin lisäksi voi oppia eri aineiden sisältöjä. Opinnäytetyön myötä syntyi Pixl Games ja sen visuaalinen identiteetti. Nimeä ja identiteettiä hyödyntämällä olen ajatellut tehdä opinnäytetyön Scratch-ohjelman ja siihen liittyvän aineiston Python ohjelmointikielelle.

4.3 Oma oppiminen

Opinnäytetyön toiminnallisen osan laatimisen myötä opin Scratch-ohjelmointiympäristön ja kuinka Scratch-ohjelmia kirjoitetaan. Samalla opin kuinka ohjelman lähdekoodista laaditaan vaiheittain eteneviä ohjelmointitehtäviä tehtävävihkoon. Syntyi prosessi, jota hyödyntämällä uusia ohjelmia ja niihin liittyviä ohjeita on nopea toteuttaa. Prosessin aikana tutustuin moniin ilmaisiin työkaluihin joiden avulla tämänkaltaisia oppimateriaaleja voi toteuttaa. Merkittävin oppimani asia kuitenkin oli, että Scratch on monipuolinen ja ilmaisuvoimainen työkalu.

Lähteet

Ammattinetti 2018. Ohjelmistosuunnittelija ja ohjelmoija. Luettavissa: http://www.ammattinetti.fi/ammatit/detail/328_ammatti. Luettu: 17.2018.

Bloomberg, J. 2018. Think big with microservices: Lego-like software development takes shape. Luettavissa: https://techbeacon.com/think-big-microservices-lego-software-development-takes-shape. Luettu: 18.2.2018.

Derwin. 2016. Quality Assurance is broken. Here's how we can make it as agile as everything else. Luettavissa: https://medium.freecodecamp.org/quality-assurance-is-broken-heres-how-we-can-make-it-as-agile-as-everything-else-64bd19d5e426. Luettu: 11.4.2018.

Faller, P. 2.6.2017. Making The Most Out of Audio: Designer Chris Mears on Sound in UX Design. Luettavissa: https://theblog.adobe.com/making-the-most-out-of-audio-designer-chris-mears-on-sound-in-ux-design/. Luettu: 11.4.2018.

Font Awesome. Luettavissa: https://fontawesome.com/. Luettu: 23.2.2018.

Google Fonts. Luettavissa: https://fonts.google.com/about. Luettu: 23.2.2018.

Google Slides. Luettavissa: https://www.google.com/slides/about/. Luettu: 23.2.2018.

Google Translate. Luettavissa: https://translate.google.com/. Luettu: 23.2.2018.

Halme, A. 2018. Ohjelmistokehitys. Digitalisoinnin opas. Luettavissa: https://www.itewiki.fi/opas/ohjelmistokehitys/. Luettu: 17.2.2018.

Hjelm, E. 2016. Tytöt, pojat ja tietojenkäsittelyn opettaminen. Luettavissa: http://urn.fi/URN:NBN:fi-fe2017112252231. Luettu: 2.4.2018.

Itewiki 2018. Graafinen suunnittelu. Digitalisoinnin opas. Luettavissa: https://www.itewiki.fi/opas/graafinen-suunnittelu/. Luettu: 23.2.2018.

Järvenpää, J. 2018. Raapaisu. Edukustannus. Helsinki.

Liukas, L & Mykkänen, J. 2014. Koodi 2016. Lönnberg Print. Helsinki.

Lego Group 2018. LEGO Groupin historia. Luettavissa: https://www.lego.com/fi-fi/aboutus/lego-group/the_lego_history. Luettu: 30.3.2018.

Neuman, A. 2017. Motivoiko ohjelmointi? Kuudennen luokan oppilaiden ja opettajan käsityksiä ohjelmoinnin opetuksesta ja oppimisesta. Luettavissa: http://urn.fi/URN:NBN:fi:hulib-201705104055. Luettu: 31.3.2018.

Mansikka, O. 2016. OAJ: Opettajien perehdytys uuteen opetussuunnitelmaan riittämätöntä – ”Näinä säästöjen aikoina koulutusta on kovin vähän”. Luettavissa: https://www.hs.fi/kotimaa/art-2000002925397.html. Luettu: 27.2.2018.

Material Design. Luettavissa: https://material.io/. Luettu: 23.2.2018.

Muranen, A. & Harmainen, L. 2017. Käyttöliittymä- & käyttäjäkokemussunnittelu (UI & UX Design). Digitalisoinnin opas. Luettavissa: https://www.itewiki.fi/opas/kayttoliittymasuunnittelu-ux-user-experience-design-eli-kayttajakokemus/. Luettu: 23.2.2018.

Opetushallitus 2014. Perusopetuksen opetussuunnitelman perusteet 2014. Next Print Oy. Helsinki. Luettavissa: http://www.oph.fi/saadokset_ja_ohjeet/opetussuunnitelmien_ja_tutkintojen_perusteet/perusopetus. Luettu: 17.2.2018.

Peda. Algoritminen ajattelu. Luettavissa: https://peda.net/jyu/it/koulutusteknologia/op/keos-2017/aa/moaa. Luettu: 30.3.2018.

Rahja, R. 30.10.2017. Lupa kasvaa – myös digimaailmassa. Luettavissa: https://www.mll.fi/2017/10/lupa-kasvaa-myos-digimaailmassa/. Luettu: 17.2.2018.

Raivonen, P. 2018. ”Rohkeasti vain kokeilemaan” – Tutkimus alakoulun opettajien valmiudesta opettaa ohjelmointia. Luettavissa: http://urn.fi/URN:NBN:fi:hulib-201803011374. Luettu: 2.4.2018.

Reimer, J. 23.7.2007. Scratch makes programming like playing with LEGO bricks Luettavissa: https://arstechnica.com/uncategorized/2007/07/new-educational-tool-makes-programming-like-playing-with-lego-bricks/. Luettu: 11.4.2018

Ribbonfarm Consulting. 2018. A new soft technology. Luettavissa: https://breakingsmart.com/en/season-1/a-new-soft-technology/. Luettu: 2.4.2018.

Rose, S. 2017. Software Project Team Roles and Responsibilities. Luettavissa: https://medium.com/@SherrieRose/software-project-team-roles-and-responsibilities-152a7d575759. Luettu: 17.2.2018.

Scratch. 2018. Luettavissa: https://scratch.mit.edu/. Luettu: 30.3.2018.

Scratch Wiki. 2018. Luettavissa: https://en.scratch-wiki.info. Luettu: 31.3.2018.

Sound of text. About. Luettavissa: https://soundoftext.com/. Luettu: 23.2.2018.
SumoPaint. Luettavissa: https://www.sumopaint.com/about/. Luettu: 24.2.2018.

The Scratch Team. 2018. 3 Things To Know About Scratch 3.0. Luettavissa: https://medium.com/scratchteam-blog/3-things-to-know-about-scratch-3-0-18ee2f564278. Luettu: 7.4.2018.

Ängeslevä, S. 28.8.2017. Tuotejohtaja. Unity. Haastattelu. Helsinki. Katsottavissa: http://digimama.fi/insights/. Katsottu: 17.2.2018.

Liitteet

Liite 1. Tehtävävihkon keskeisimmät osat

Ministeriö

Brief

Hei,

Olemme täällä Ministeriössä hankalan haasteen edessä. Ministeriö vastaanottaa Venäjän, Iso-Britannian ja Ruotsin valtioiden delegaation. Tapaamisen tarkoitus on miettiä eri maiden välisiä suhteita ja kuinka parantaa niitä. Aivan ensimmäiseksi tarvitsisimme teiltä prototyypin, jonkinlaisen aparaatin, jolla delegaatioon osallistuvien korkea-arvoisten henkilöiden lapset voisivat opettaa vieraita kieliä toisilleen ja sisaruksilleen.

Aparaatin prototyypin tulisi opettaa numerot nollasta kymppiin eri kielillä. Aparaatissa tulisi olla ainakin seuraavat kielet: Suomi, Ruotsi, Venäjä ja Englanti.

Odotamme teiltä selkeätä suunnitelmaa aparaatista ja sen toimintaperiaatteista. Koska suomi tunnetaan teknologian edelläkävijä maana, meillä on suuret odotukset luovan teknologian osalta.

Ystävällisin terveisin,
Ministeriön Kansliapäällikkö

Ministeriö

NUMEROT - SIFFRORNA

Ruotsi - Svenska

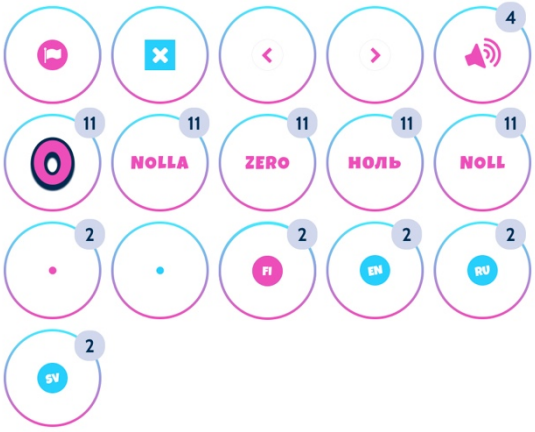
0. noll

1. en
2. två
3. tre
4. fyra
5. fem
6. sex
7. sju
8. åtta
9. nio
10. tio

PIXL GAMES

SPRITES

Hahmot



Petri Santala © 2018

PIXL GAMES

BACKDROPS

Taustat

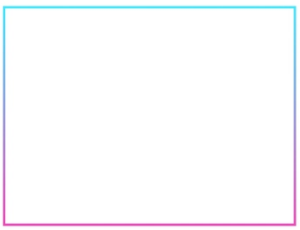
1.

backdrop-exit

Aloitusnäkymän tausta, joka näytetään myös silloin, kun ohjelma lopetetaan.

2.

backdrop



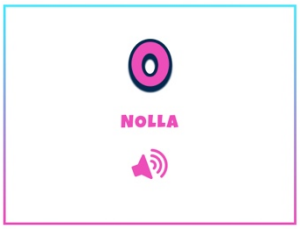
Tausta, joka näytetään ohjelman ollessa käynnissä.

Petri Santala © 2018

PIXL GAMES

KÄYTTÖLIITTYMÄSUUNNITTELU

Piirrä mallin mukaan

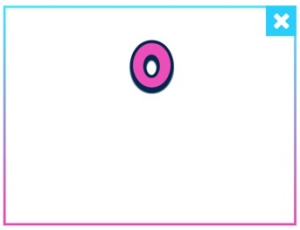


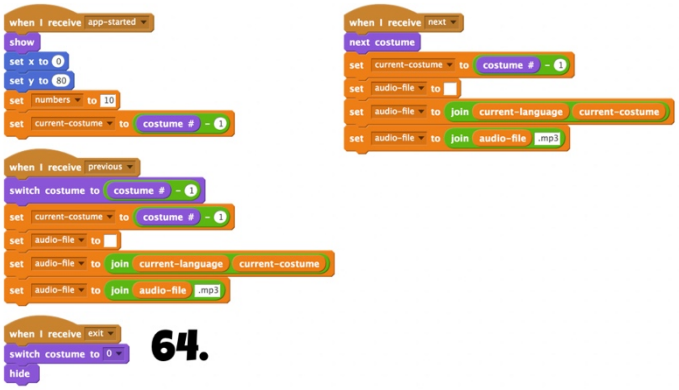
Petri Santala © 2018

PIXL GAMES

NUMBERS

Ohjelmoi





64.

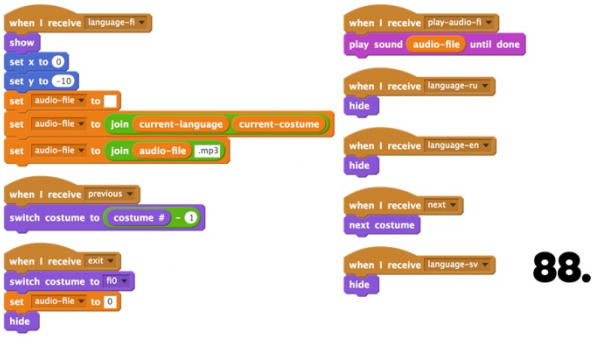
Petri Santala © 2018

PIXL GAMES

NOLLA

KIELI – SUOMI – FI

Ohjelmoi



88.

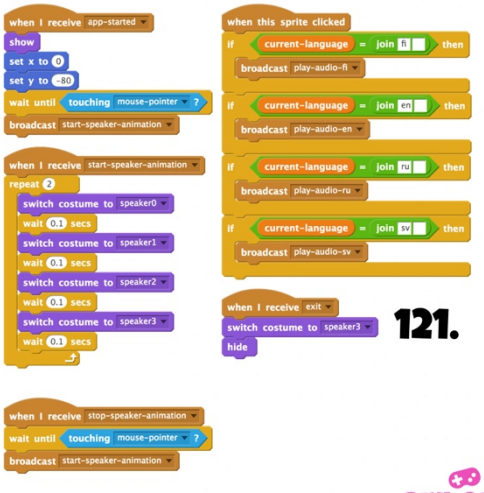
Petri Santala © 2018

PIXL GAMES

NOLLA

AUDIO

Ohjelmoi selostus ja kaiutin-painikkeen animaatio



121.

Petri Santala © 2018

PIXL GAMES